Pervasive and Mobile Computing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Fast track article

Dynamic address autoconfiguration in hybrid ad hoc networks

Emilio Ancillotti, Raffaele Bruno *, Marco Conti, Antonio Pinizzotto

Institute for Informatics and Telematics (IIT) - Italian National Research Council (CNR), Via G. Moruzzi, 1 - 56124 Pisa, Italy

ARTICLE INFO

Article history:
Received 10 March 2008
Received in revised form 29 July 2008
Accepted 23 September 2008
Available online xxxx

Keywords:
Ad hoc networks
WLAN
Address auto-configuration
DHCP
Measurements

ABSTRACT

The use of ad hoc networking technologies is emerging as a viable and cost-effective solution to extend the range of traditional wireless local area networks (WLANs). In these networks, mobile client traffic reaches the access points through multi-hop wireless paths that are established by using an ad hoc routing protocol. However, several technical challenges have to be faced in order to construct such an extended WLAN. For instance, traditional autoconfiguration protocols commonly used in infrastructure-based WLANs, such as DHCP or Zeroconf, are not directly applicable in multi-hop wireless networks. To address this problem, in this paper we propose extensions to DHCP to enable the dynamic allocation of globally routable IPv4 addresses to mobile stations in hybrid ad hoc networks, which transparently integrate conventional wired technologies with wireless ad hoc networking technologies. Some of the attractive features of our solution are its ability to cope with node mobility, the introduction of negligible protocol overheads, and the use of legacy DHCP servers. We have implemented a prototype of our scheme, and tested its functionalities considering various topology layouts, network loads and mobility conditions. The experimental results show that our solution ensures short address configuration delays and low protocol overheads.

1. Introduction

In recent years, we have witnessed an exceptional growth of the number of deployed wireless local area networks (WLANs) as a result of the commercial success of the IEEE 802.11 technology [1], and the consequent increase in the number of wireless users. A typical 802.11-based WLAN consists of two different entities: access points (APs), also called base stations, which are connected to the network infrastructure, and mobile clients (or stations), which are associated with an AP that is reachable through single-hop wireless transmissions. However, due to radio signal attenuation, the coverage area of a single WLAN is quite limited. In addition, several factors such as electromagnetic interference, fading, obstacles, etc., may impair the radio transmissions. For these reasons, ensuring truly seamless network coverage to a mobile user can be a challenging task.

To extend the range of WLAN systems, two approaches are traditionally followed in real practice. On the one hand, it would be possible to increase the transmission power of an access point in order to reach farther nodes. However, the main shortcoming of this solution is that it may lead to a poor channel reuse because a larger number of users should access the network through the same base station. Consequently, the contention level within each cell increases, thus degrading the per-client throughput. Moreover, the effectiveness of this technique is limited by the fact that the IEEE 802.11 technology operates in an unlicensed frequency spectrum (i.e., the ISM band) [1], and national regulations usually set stringent limits to the maximum transmission-power levels in unlicensed bands. Alternatively, we may opt for deploying more access points at a closer spacing, increasing the network capacity. However, a number of reasons, including co-channel interference between

* Corresponding author.
E-mail addresses: a.ancillotti@iit.cnr.it (E. Ancillotti), r.bruno@iit.cnr.it (R. Bruno), m.conti@iit.cnr.it (M. Conti), a.pinizzotto@iit.cnr.it (A. Pinizzotto).

1574-1192/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2008.09.008

nearby access points, availability of a limited number of orthogonal non-interfering frequency channels, as well as cost and management overheads, limit the effectiveness of this alternative solution.

To overcome the limitations of the above-discussed approaches, several authors have recently advocated a new architecture for WLANs, which integrates ad hoc networking technologies in the network infrastructure [2–5]. Traditionally, mobile ad hoc networks (MANETs) are conceived as an isolated collection of mobile nodes connected together over a wireless medium, which self-organize into an autonomous multi-hop wireless network [6]. However, it is now recognized that the ad hoc networking paradigm can also be applied to infrastructure-based wireless networks, building a *hybrid ad hoc network*, and providing a flexible, robust and cost-effective increase of network coverage. Specifically, we envisage an *extended WLAN* in which static and mobile clients transparently communicate using traditional wired technologies or ad hoc networking technologies. Thus, the client traffic can be forwarded to the access points through multi-hop wireless paths established by using an ad hoc routing protocol [5]. It is important to underline that other classes of hybrid ad hoc networks have emerged from this vision, such as: Multi-hop Cellular Networks (MCN), which combine the features of cellular systems and ad hoc networks [7], and mesh networks, which employ a multi-hop wireless backbone to provide Internet access to mobile users [8].

Several technical challenges have to be faced in order to construct such a hybrid ad hoc network because the characteristics of the ad hoc networking (e.g., multi-hop relaying, lack of a centralized administration, etc.) differ significantly from the conventional IP architecture. For instance, the address autoconfiguration protocols commonly used in infrastructure WLANs, such as the Dynamic Host Configuration Protocol (DHCP) [9] or the Zeroconf protocol [10], are not directly applicable in multi-hop wireless networks. However, a mobile device cannot participate in unicast communications until it has been assigned a free IP address and the corresponding subnet mask. It is evident that pre-configuration is impractical in mobile environments, as well as a violation of the self-organizing paradigm. Thus, an address autoconfiguration protocol is crucial to allow the dynamic and automatic allocation of unique IP addresses to mobile clients. To tackle this problem we propose extensions to DHCP to enable the automatic allocation of globally routable IPv4 addresses to mobile stations in the envisaged extended WLAN.[1] Important features of our proposed solution are the following: (i) it is a fully distributed and automatic scheme that does not maintain state information in the already configured nodes, (ii) it does not assume that the address allocation space is known a priori by the new nodes, (iii) it does not require changes of the legacy DHCP-server implementation, (iv) no DHCP servers are deployed in the ad hoc component of the extended WLAN (see Section 3 for a detailed description of the network architecture), (v) it is designed to efficiently cope with node mobility, and (vi) it generates negligible and controlled protocol overheads. Note that DHCP is usually considered not applicable to MANETs since, in case the DHCP server is running on a mobile node, the DHCP server might not be permanently reachable by all nodes. However, our solution is not affected by this problem, since new nodes communicate directly with the DHCP servers deployed on the wired part of the extended WLAN by exploiting the relay capabilities of already configured nodes.

In principle, it may be argued that any other autoconfiguration protocol proposed for ad hoc networks might be also employed to assign a unique network-layer identifier to mobile stations in the envisaged extended WLAN. However, autoconfiguration protocols for MANETs are generally designed to select an identifier with a scope limited to the ad hoc network [12]. This approach is reasonable for *stand-alone* MANETs, which are not connected to external networks, but it introduces additional complexities once we permit the interconnection between ad hoc networks and the Internet. Specifically, if private IP addresses are used within the MANET, a network address translator (NAT) has to be implemented on each gateway to enable IP communications. Then, the NAT-based gateway translates the source private IP address of outgoing traffic with a globally valid IP address, which is routable on the Internet. However, recent studies have clearly demonstrated that NAT-based gateways are very inefficient when multi-homing (i.e., more than one gateway in the same MANET) is allowed and the network topology is highly dynamic [5,13,14]. On the contrary, in our previous paper [5] we have shown that the use of globally routable IP addresses in the ad hoc network permits to implement very efficient gateways that support transparent IP communications, even in highly mobile conditions. These observations motivate our efforts to use DHCP for assigning globally valid IP addresses also to ad hoc nodes. Note that an alternative approach to configure globally routable IP address would be to use a hardware-based addressing. In other words, a global network prefix may be assigned a priori to the ad hoc network, and the IP address is then completed using the node's unique hardware interface identifier. However, this approach requires additional features that are only available in IPv6. In addition, it is not always true that network interfaces have globally unique addresses, but violations of this assumption are possible.

To verify if our scheme guarantees satisfactory configuration delays and an acceptable efficiency in terms of protocol overheads, we have implemented a fully operational prototype and we have tested its functionalities, taking into consideration various topology layouts, network loads and mobility conditions. Our experimental results show that: (i) even if the new client is several hops far from the DHCP server, and asymptotic TCP flows saturate the wireless links, the configuration delays are acceptable, and (ii) the protocol overheads are negligible even if node mobility interferes with the operations of the autoconfiguration protocol.

The remaining of this paper is organized as follows. Section 2 outlines the related work on address autoconfiguration protocols for MANETs. In Section 3 we define the architecture of an extended WLAN. Section 4 briefly reviews the DHCP specification. The basic idea of the proposed solution is presented in Section 5, while the protocol details are described in Section 6. Section 7 presents the experimental evaluation, and Section 8 concludes the paper with final remarks.

[1] An initial version of our proposal, as well as preliminary experimental results, were presented in [11].

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

2. Related work

Various address autoconfiguration protocols for MANETs have been proposed in the literature, and it is out of the scope of this paper to present a complete review. Rather, we focus on outlining the various approaches that have been adopted and the features of representative solutions. The reader is referred to [12] for an exhaustive survey.

Generally speaking, autoconfiguration protocols for ad hoc networks can be classified as *stateless*, *stateful* or *hybrid* solutions. Protocols following a stateful approach are very structured schemes, because every node has to maintain detailed state information about the utilization of the MANET address space. This state information is usually represented by an address allocation table that contains the addresses currently in use within the ad hoc network. The main challenge of this class of solutions is the maintenance of the allocation table consistency, especially in the presence of packet losses and network merging. One of the first schemes employing a stateful approach with a distributed allocation table is the MANETconf [15] protocol. With MANETconf an unconfigured node selects a reachable MANET node as the *initiator* of the address allocation procedures. The initiator selects an address that has not been used yet (at least according to its local address table), and it broadcasts a request containing this address to all the nodes in the MANET. An allocation is assumed to be successful only if the initiator receives a positive reply from all the nodes in the MANET. Note that, due to message unreliability, inconsistencies in the allocation tables are still possible, and this may lead to unnecessary address changes or undetectable conflicts. To ensure reliable global synchronization of the allocation tables, it is fundamental to implement reliable broadcast mechanisms, which are generally complex and resource-consuming protocols. To avoid maintaining complete allocation tables in each node, which may not scale in large MANETs, the Prophet protocol [16] follows a different approach. Specifically, each node in the MANET maintains a function $f(n)$ and a state value, called *seed*, to generate a sequence of integers. Function $f(n)$ is chosen in such a way that the probability to select the same integer when different *seeds* are used is extremely low. When a new node, say *B*, wants to join the MANET it broadcasts an address request to one of its neighbors, say *A*, which selects a new *seed* and generates an integer applying this *seed* to $f(n)$. Then, node *B* will use the generated value as its IP address, and the state value obtained from *A* as the seed to assign IP addresses to other new nodes. Note that this protocol may generate duplicate addresses. Thus, additional mechanisms are needed to detect and solve these conflicts.

In principle, stateless protocols are less complex solutions than stateful schemes, because each node selects autonomously its own address and performs a Duplicate Address Detection (DAD) procedure to verify its uniqueness and resolve conflicts. However, Perkins et al. [17] proposed one of the first schemes by adapting the IETF Zeroconf protocol to the MANET case. The basic idea is that each new node selects a random address from a *pre-configured address space*. This means that the IP address block from which nodes have to choose their IP addresses is known in advance to each node. After self-assigning an IP address from this allocation address space, the new node queries all other nodes in the ad hoc network to verify if one of them is already using this address. If the new node does not receive any negative reply within a given timeout and after multiple tries, it will assume that the chosen address is not currently used in the MANET. Two drawbacks can be identified in this scheme. The first one is unreliability caused by the exclusive use of timeouts to stop the DAD procedure, because message delays may be unbounded in an ad hoc network. The second one is the protocol overhead generated by the flooding of the network with address request messages. To increase the protocol efficiency, a different strategy is described in [18], called *weak* DAD, which integrates the DAD mechanism with the routing protocol. More precisely, each node generates a key at initialization time (either randomly or based on an unique hardware ID) and distributes this key in the routing messages. Duplicate addresses are detected by receiving packets with an address that corresponds to multiple keys. Nevertheless, conflicts cannot be detected if two nodes select the same key and the same address. This event is unlikely if the key length is sufficiently large. However, increasing the key length also increases the routing protocol overheads. An optimization of this approach is proposed in the PACMAN (Passive Autoconfiguration for Mobile Ad Hoc Networks) protocol [19], where no additional information (i.e., keys) is sent in the routing node messages, but every node analyzes the routing traffic to identify anomalies. Thus, this protocol implement a *passive* DAD mechanism because conflicts are detected by passively awaiting for routing events that would not have occurred with unique addresses. PACMAN can be classified as a hybrid scheme because every node maintains also an address allocation table. However, these tables are not synchronized, and an unconfigured node can request the allocation table from neighboring nodes only to expedite the configuration process. A shortcoming of this approach is that the DAD procedure depends on the specific routing protocol used in the MANET.

Before concluding this review of related work, it is also useful to outline the activities of the IETF AUTOCONF working group [20], which is studying the standardization of mechanisms for configuring unique local and/or globally routable IPv6 addresses. In principle, IPv6 should make the autoconfiguration of unique addresses easier than IPv4 because the size of the IPv6 address space permits each node to build its own globally routable IPv6 address by embedding a globally unique hardware ID (e.g., the 48 bit IEEE MAC address). This is the basic idea of the original IPv6 stateless address autoconfiguration protocol [21], and its extension to the MANET case [22]. However, no hardware ID can be considered really globally unique. For instance, interface drivers permit to dynamically change the MAC address. For these reasons, the proposals that have received more attention in the research community are the schemes that use gateway nodes to distribute within the MANET a network prefix that can be used for configuring a (typically globally) routable IPv6 address. One solution is described in [23], which defines both proactive and reactive strategies to discover the gateways within the ad hoc network. An alternative solution is described in [24]. This scheme introduces the concept of "prefix continuity". More precisely, multiple subnets (i.e., network prefixes) can be used in the same MANET. However, network identifiers should be assigned to visiting nodes

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

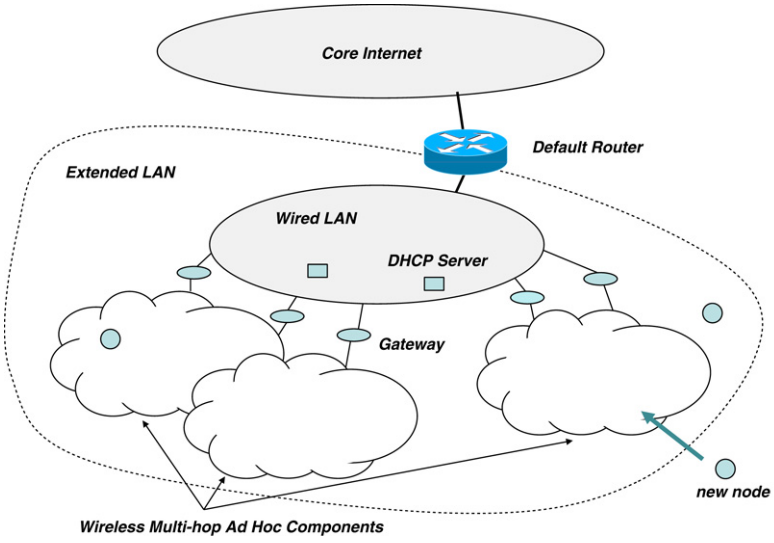


Fig. 1. Reference network architecture.

in such a way that any node has at least one neighbor using the same prefix. In other words, the MANET should be organized in clusters of hosts sharing the same network prefixes. This network organization reduces the overheads introduced by flooding gateway advertisements.

3. Network model

Before describing the details of the proposed extensions to DHCP, it would be useful to illustrate the complete network architecture we consider for building hybrid ad hoc networks interconnected to the Internet. The application scenario we envisage for this system consists in providing a cost-effective, seamless and robust wireless Internet access for nomadic users in small-scale areas, such as campuses or enterprise buildings. The design goal is to ensure transparent communications between static hosts, which use traditional wired technologies, and mobile clients, which use more advanced ad hoc networking technologies [5]. For the sake of clarity, in Fig. 1 we depict the reference network architecture we consider in our study.

As illustrated in the figure, we envision an extended WLAN, hereafter also indicated as *multi-hop WLAN*, composed of a conventional LAN (the wired component) and several ad hoc components. In this network mobile clients not in close proximity to the fixed networking infrastructure establish multi-hop wireless paths to communicate with each other using an ad hoc routing protocol. Special devices, named gateways, interconnect the wired LAN with the ad hoc components. These gateways are static devices with multiple interfaces. One fixed interface is used to connect the gateway to the wired LAN, while the other wireless interface operates in ad hoc mode. Thus, a gateway can be seen as an enhanced access point supporting ad hoc networking, rather than infrastructure-based wireless communications. Finally, standard IP routing is used to connect the extended WLAN to the core Internet.

In our architecture multi-homing is permitted, i.e., multiple gateways can be located within the same ad hoc component, and the ad hoc routing manages the network-layer handoff of mobile nodes between gateways.[2] We assume that DHCP servers are deployed in the wired component to administer the dynamic assignment of unique IP addresses to both wired host and mobile clients temporarily associated to the network. This ensures that the *the extended WLAN is a single address space*, where both ad hoc and static hosts have an IP address with the same network identifier. In our previous work [5] we have shown that this architectural design allows transparent support for node mobility and facilitates Intranet communications. In the following sections we describe how an unconfigured mobile host that wants to join the multi-hop WLAN, for brevity denoted as *new node*, can query the DHCP servers to obtain its IP configuration parameters.

4. DHCP standard

In this section we outline the DHCP specification for IPv4, i.e., DHCPv4 [9]. Note that the modifications introduced with IPv6 to the original IP addressing architecture required a complete redesign of the DHCP standard [25]. Thus, the extensions to DHCP proposed in this paper are applicable only to DHCPv4.

[2] Note that in ad hoc mode there is not link-layer handoff because mobile nodes does not have to *associate* to the gateways.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

DHCPv4 (hereafter simply DHCP) is designed exploiting the client/server model, and DHCP clients and servers interact through a series of client-initiated request–response transactions. Obviously, the DHCP server plays a central role in DHCP because it provides the configuration parameters to the Internet hosts (clients) that communicate with it. In small networks it can be sufficient a single server to support many clients, while large networks may require multiple DHCP servers. The DHCP servers are the owners of the addresses used by all DHCP clients and manage their use, keeping track of both the allocated addresses and the available ones. The most efficient mechanism used by DHCP servers for assigning IP addresses is the *dynamic allocation mode*, which provides a time-limited address allocation. Specifically, DHCP servers assign IP addresses to clients on a lease, and, before the lease expires, DHCP clients should request the renewal of the lease. In this way DHCP servers can immediately reuse IP addresses that have not been renewed.

Generally speaking, the DHCP communication protocol consists of responses issued by one or more DHCP servers in reply to different types of requests from clients. To describe the client–server interactions it is useful to give an example of a typical dynamic address allocation. First, when a DHCP client boots up, it sends a DHCP_Discover packet to its local physical subnet to locate available servers. This message is a layer-2 broadcast, i.e., the destination MAC address is FF:FF:FF:FF:FF:FF. Each DHCP server receiving this broadcast should respond with a DHCP_Offer sent to the client's MAC address. The DHCP_Offer includes a tentative IP address for the client, the IP address of the DHCP server sending the response, and the lease duration. A DHCP client may receive multiple DHCP_Offer messages from different DHCP servers, and the client must choose one of the servers that replied. Then, the DHCP client broadcast a DHCP_Request message to inform all the DHCP servers that the offer has been accepted. To this end, the DHCP_Request message contains the IP address of the selected DHCP server. Only that DHCP server is allowed to respond to the request message with a DHCP_ACK, which contains the rest of the information needed by the client to start conventional IP-based communications, including the location of a DNS server and a default Internet gateway. The DHCP servers that made the offers that were not accepted will return the offered IP address to their range of assignable addresses.

Since DHCP uses the unreliable User Datagram Protocol (UDP) for encapsulating messages, it defines a retransmission strategy to cope with message losses. Note that DHCP clients are responsible for detecting message losses and for all message retransmissions. Specifically, the clients adopt a retransmission strategy that incorporates a randomized exponential backoff algorithm to determine the delay between retransmissions. In general, the delay between retransmissions is doubled up to a maximum of 64 s, while the delay before the first retransmission should be 4 s plus a random value uniformly selected in the range $[-1, 1]$ [9].

One of the most important limitations of DCHP, which is common to several host configuration protocols, is the reliance on broadcasts for communication. For performance reasons, broadcasts are normally propagated only within a local network segment, and this means that DHCP clients and DHCP servers on different physical network segments cannot communicate directly. To eliminate the necessity of having a DHCP sever on every single physical subnet, a router (or a normal Internet host) can be configured as a *DHCP Relay Agent*. A DHCP relay agent will intercept DHCP_Discover and DHCP_Request packets from clients. Then, the DHCP relay can either rebroadcast the clients' DHCP messages to other networks, or send them directly to specific DHCP servers it was configured to contact. The DHCP server responds back to the relay agent that, in turn, forward the servers' replies directly to the original client's MAC address.

5. Outline of the idea

The goal of our autoconfiguration scheme, called *Ad Hoc DHCP* (*AH-DHCP*), is to assign a globally routable IPv4 address to the mobile nodes of a multi-hop WLAN using the DHCP-based mechanisms already implemented in the wired part of the network, without requiring any change of the standard DHCP server implementation. In this way we can assign globally routable IP addresses to ad hoc nodes without requiring that a pre-configured IP address space is reserved to the ad hoc components.

To enable a new node to deliver its address request to the available DHCP servers, we exploit the DHCP relay capability. More precisely, a new node should execute a preliminary discovery procedure to identify other wireless nodes already associated with the multi-hop WLAN and reachable through one-hop wireless transmissions. Then, the unconfigured node elects one of the discovered neighbors to act as DHCP relay agent, which will forward all the client's DHCP messages to the known DHCP servers. The DHCP standard does not define any specific mechanism to discover the available DHCP relay agents, but client-originated DHCP packets are implicitly forwarded by the relay agents located on the same physical network segment of the client. This behavior is acceptable in wired networks because they are controlled environments, and both the location and number of DHCP relay agents are carefully planned. Typically, DHCP relay agents are enabled only on the interfaces of routers interconnecting different subnets. On the contrary, in a multi-hop WLAN each wireless node is a potential DHCP relay agent that may act as a proxy during the configuration process of a new node. Therefore, if multiple DHCP relay agents are used concurrently to pass client's messages to DHCP servers, the DHCP servers may be overloaded by the simultaneous requests. Moreover, multiple copies of the same DHCP messages will travel in the multi-hop WLAN increasing the protocol overheads.[3] In conclusion, introducing a DHCP relay agent discovery mechanism can introduce a twofold benefit. Firstly, it reduces the number of messages generated during the configuration process.

[3] An analysis of the use of the overhead associated to the use of multiple relays is reported in Section 7.3.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

Secondly, it guarantees that the DHCP servers receive a single address request from each new node joining the multi-hop WLAN.

There is another shortcoming in the original design of DHCP that prevents its efficient use in multi-hop WLANs. Specifically, DHCP standard assumes that nodes are static during a client–server transaction, and message losses are infrequent. For these reasons, DHCP clients adopt a simple retransmission strategy that relies on timeouts to detect messages losses [9]. However, a multi-hop WLAN is a dynamic environment where nodes are free to move almost arbitrarily. Thus, the selected DHCP relay and the unconfigured node may move out of their respective transmission ranges and become unreachable before the address assignment is completed. This may lead to unacceptable delays in the address allocation. Moreover, external interference or routing protocol inconsistencies can produce not negligible packet errors. Consequently, efficient procedures should be devised to cope with node mobility, and unexpected communication problems. To this end, our scheme incorporates a mechanism to allow a timely detection of nodes' movements and/or failures in order to ensure a prompt re-selection of a new valid DHCP relay agent.

After the completion of the initial configuration procedure, each wireless node has to periodically interact with the DHCP server to renew its address. Some authors [12,15] observed that it might be difficult to guarantee a continuous access to DHCP servers since ad hoc networks can become partitioned due to node mobility. However, in the considered network scenarios this limitation does not appear problematic. First of all, the multi-hop WLAN we envision will be mostly used as a flexible and cost-effective extension of the fixed networking infrastructure in enterprise buildings or campus facilities. In these contexts, users are semi-static or nomadic and are interested in having a continuous access to Internet and its centralized services (e.g., web browsing, access to centralized data repositories, etc.). In addition, DHCP servers are located only in the wired part of the network. Thus, until the wireless node is able to reach an access point through a multi-hop path, it will be able to contact the DHCP server for address renewals.

6. AH-DHCP description

We assume that the gateways are the first nodes to join the multi-hop WLAN. Note that the gateways can interact with the DHCP servers using their wired interfaces. For this reason, AH-DHCP does not need an initialization procedure, which, on the contrary, is an important task of autoconfiguration protocols for stand-alone MANETs [15]. Thus, in the following we only describe the AH-DHCP operations when a new node (other than the access point) wants to join the multi-hop WLAN. For brevity, and whenever ambiguity does not occur, we refer to AH-DHCP clients and AH-DHCP relay agents simply as clients and relays. For the sake of clarity, in Figs. 2 and 3 we illustrate the protocol state machines of a client and a relay agent, respectively. In these diagrams we represent the events that initiate a transition in brackets (e.g., the expiration of a timeout, the reception of a specific message, etc.). If a message is generated at the end of a transition, it is represented with a box at the end of the transition arch. Furthermore, Tables 1 and 2 list the messages and parameters specific to AH-DHCP.

6.1. DHCP relay discovery phase

Let node C be a new mobile node that wants to join the multi-hop WLAN. To this end, it has to query a DHCP server for receiving the necessary IP configuration parameters. Thus, node C starts its AH-DHCP client module entering into the "DHCP relay discovery" state. Then, node C periodically broadcasts special messages, called RELAY_DISCOVER messages (see Fig. 2), with period T_R. Every wireless node that is already part of the multi-hop WLAN, and is running a relay agent, after receiving a RELAY_DISCOVER message, should reply with a RELAY_ACK message (see Fig. 3). This RELAY_ACK message expresses the willingness of the relay agent to act as *initiator* of the address configuration process for node C. Note that RELAY_DISCOVER messages are broadcast frames that can be received only if two nodes are in radio visibility, while RELAY_ACK messages are unicast frames sent directly to node C's MAC address.

Each RELAY_ACK message transports a list of attributes characterizing the DHCP relay capabilities, such as the remaining battery energy, the distance (in terms of hops) between the relay and its closest gateway, if the relay is already involved in a configuration procedure for another wireless node, etc. The identity (MAC and IP address) and the attributes of each relay that replied to a RELAY_DISCOVER message are stored in a temporary cache, called *relay_cache*. Node C allocates a fixed time, say T_O, to collect the neighbors' responses. After the expiration of this timer, node C selects the "best" relay according to a pre-defined policy applied to the attributes of discovered relays. In our prototype we have implemented the following strategy: the relay that is at the minimum distance from an access point should be selected as the forwarder of DHCP messages. Note that other mechanisms could be devised to select a single DHCP relay agent. For instance, it could be possible to implement a timer-based response mechanism where the timer is set based on some preferences (e.g. DHCP relays having a shorter distance from the gateways have smaller timeout values, and may respond first). However, a DHCP relay discovery scheme based on period broadcast messages can be easily integrated into classical hello-like neighbor discovery schemes implemented in popular ad hoc routing algorithms (e.g., AODV or OLSR). After selecting a DHCP relay agent, say R_A, node C can begin a conventional DHCP transaction by sending a unicast DHCP_DISCOVER message to R_A. Note that legacy DHCP clients transmit broadcast DHCP_DISCOVER messages because they are not aware of the available DHCP relays. On the contrary, since AH-DHCP clients scan their neighborhood to discover available AH-DHCP relays, they can use a single relay as unique

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

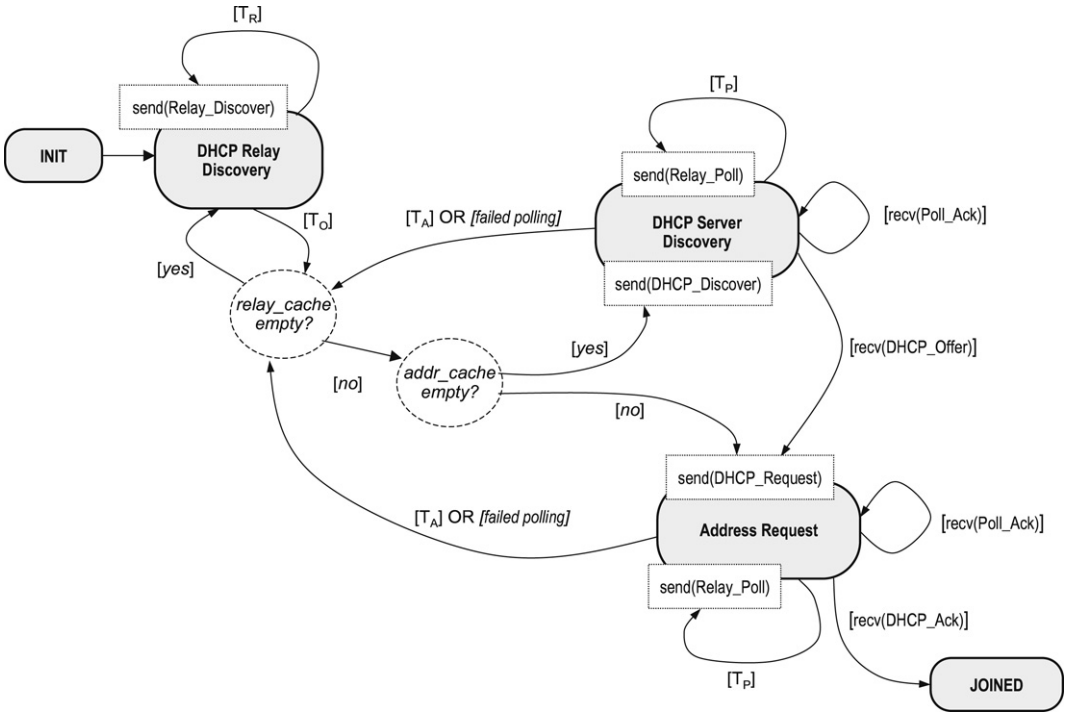


Fig. 2. State machine of the AH-DHCP client's behavior.

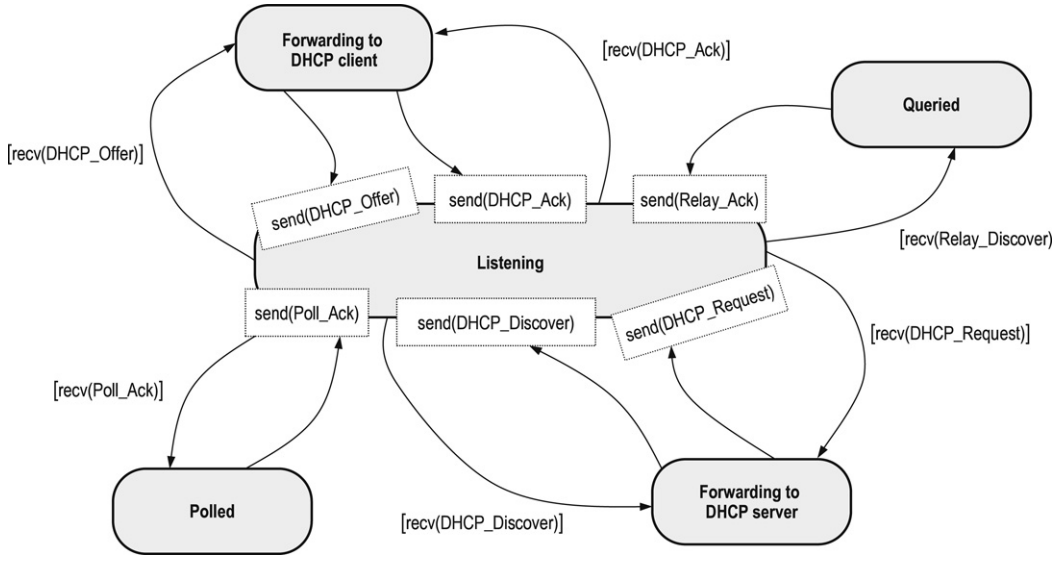


Fig. 3. State machine of the AH-DHCP relay agent's behavior.

Table 1
AH-DHCP message notation.

Message type	Message description
RELAY_DISCOVER	Hello-like message sent by a new node during the DHCP relay discovery phase
RELAY_ACK	Reply of DCHP relays to RELAY_DISCOVER messages
RELAY_POLL	Poll message sent by a new node to the selected DHCP relay agent
POLL_ACK	Reply of DHCP relays to RELAY_POLL messages

initiator of the address allocation process. This avoids sending multiple copies of the same allocation request to the DHCP servers.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

Table 2
AH-DHCP parameters.

Parameter type	Parameter description	Default value
T_R	Repetition period of RELAY_DISCOVER messages	20 ms
T_O	Maximum duration of DHCP relay discovery phase	100 ms
T_P	Repetition period of RELAY_POLL messages	50 ms
$\max_{\text{miss}}$	Relay_Poll	4
T_A	Timeout for a DCHP transaction	3 s

As described above, to increase the probability of receiving at least a response from neighboring wireless nodes, node C periodically broadcasts new RELAY_DISCOVER messages with period T_R. However, to avoid synchronization with other AH-DHCP clients in radio visibility of node C and transmitting RELAY_DISCOVER messages, the generation of these packets should be randomized. Several randomization schemes have been proposed in literature for wireless environments, especially for multi-hop broadcasting [26]. However, during the DHCP relay discovery phase we use only local broadcasts to discover one-hop neighbors. Thus, we may avoid sophisticated mechanisms to reduce collision probability. For the RELAY_DISCOVER messages, we simply add a variable jitter to the time instant at which a new message should be transmitted. More precisely, if t_k is the time instant at which node C should transmit the kth RELAY_DISCOVER message, the real transmission is scheduled at time $t'_k = t_k + jitter$, where *jitter* is a random value selected in the interval $[-\text{MAX}_j, \text{MAX}_j]$. In our prototype implementation we selected $\text{MAX}_j = 0.1 \cdot T_R$. Note that this randomization strategy is similar to the one adopted in the OLSR specification [27] to avoid synchronization of routing control messages. Similarly, it is possible to have collisions involving the RELAY_ACK replays, because node C may have a large number of neighboring nodes with DHCP relaying capabilities. Again, we adopt as collision avoidance strategy the randomization of RELAY_ACK transmissions, but we provide to these packets a higher level of spreading by selecting a maximum jitter value equal to 50% of T_R. Finally, it is possible that after the T_O expiration, node C has not received any response. In this case, node C re-initializes the T_O timer and continues transmitting RELAY_DISCOVER messages.

6.2. DHCP transaction

After sending the unicast DHCP_DISCOVER message to the selected relay R_A, node C waits in "DHCP server discovery" state for receiving a DHCP_OFFER message from the DHCP sever, which the relay agent has forwarded the message to (see Fig. 2). As explained in the Section 4, each DHCP_OFFER message contains the tentative configuration parameters offered by the replying DHCP server. Thus, node C, after receiving a DHCP_OFFER message, extracts these configuration parameters and store them in a temporary cache, called *addr_cache*. Then, node C sends a unicast DHCP_REQUEST (note that in standard DHCP, DHCP_REQUEST messages are broadcast messages) to the selected relay, and it waits in the "Address request" state for receiving a final DHCP_ACK message from the DHCP sever, which would complete the configuration process. When node C has received from the DHCP server the confirmation for using the requested configuration parameters, it can start the ad hoc routing agent and get associated to the multi-hop WLAN. It also activates its internal DHCP relay agent to intercept the requests of future nodes that want to join the multi-hop WLAN.

It is important to note that in our solution the number of DHCP messages received by the DHCP server is constant, and independent of the network topology. Furthermore, the number of DHCP messages transmitted in the network during a DHCP transaction depends only on the number of hops of the shortest path between node C and its closest gateway. For instance, let us assume that node C has n neighboring DHCP relays, and that the closest gateway is $d + 1$ hops far from node C. Under the hypotheses that no DHCP messages are lost during a DHCP transaction, it is straightforward to derive that the number of DHCP messages transmitted in the ad hoc network is equal to $4d$ (a DCHP transaction is composed of four DHCP messages, which are replicated on each of the d links between the DHCP relay and its closest gateway), while the DHCP server receives only two DHCP messages and generates two replies (see Section 7.3 for experimental results confirming these observations). On the contrary, activating all the n available relays generates *uncontrolled* overheads, and an excessive number of messages per-DHCP transaction. More specifically, the number of DHCP messages transmitted in the ad hoc network is at least $4d \cdot n$. Note that this is a lower bound for the protocol overhead because some of the DHCP relays may have their closest gateway further than d hops. Moreover, the DHCP server will receive $2n$ DHCP messages, generating $2n$ replies. In other words, the protocol overheads increases at least linearly with the number of neighbors of node C.

As noted in Section 4 each node has to periodically renew its DHCP lease with the DHCP server. However, it may happen that the DHCP relay agent is not able to contact the DHCP server (e.g., due to inconsistencies of routing table, poor link qualities, etc.) and to renew its IP network parameters. In this case the node cannot participate to the routing because its IP information have to be considered stale. Thus, this node has to repeat the address autoconfiguration process described in Section 6.1 to acquire new IP network parameters. Nevertheless, it is reasonable to believe that the failure of a renewal attempt will be a rare event, with no appreciable impact on the configuration latencies of new nodes joining the network.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

6.3. Message losses and local node mobility

In the previous section we have implicitly assumed that there are no DHCP message losses. However, in real environments DHCP messages can be lost for several reasons. For instance, it can occur that between the selected relay and the access point there are persistent communication problems (e.g., overloaded channels, link breakages, etc.) that make the transmission delays unlimited. In addition, frames can be lost due to channel interference or unexpected node crashes. Finally, being mobile, node C and the selected relay R_A can move during the DHCP transaction without remaining in radio visibility. As explained in Section 4, legacy DHCP clients implement a retransmission strategy using a randomized exponential backoff algorithm, with a maximum retransmission delay of 64 s [9]. Such a delay is acceptable only because DHCP message losses are assumed extremely rare in wired networks. However, this strategy is not adequate to cope with a highly dynamic system. To ensure that node C is able to promptly discover a topology change, we implement a *proactive pollling* mechanism in our AH-DHCP client. Specifically, during a DHCP transaction the new node C sends periodic unicast RELAY_POLL messages, with period T_P, to the selected DHCP relay R_A, which mandatorily replies with a POLL_ACK message. If R_A does not reply to $\max_{\text{miss}}$ consecutive polls, node C can assume that relay R is not reachable anymore and it removes that relay from the *relay_cache*. Note that R_A stops replying to node C's RELAY_POLL messages also if it looses its connection to the gateway. The generation period of RELAY_POLL messages and the $\max_{\text{miss}}$ value should be chosen as a tradeoff between the promptness in detecting topology changes, protocol overheads and the tolerance to poll message losses. As shown in Section 7.2, with a proper setting of the polling mechanisms, the increase of address configuration latency due to node mobility can be of the order of a few tens of milliseconds in some configurations.

After a failed polling, node C should search an alternative relay in its *relay_cache* (see Fig. 2). However, if no alternative relays are already known, the only choice for node C is to start a new DHCP relay discovery phase. On the other hand, if an alternative DHCP relay is known, say R_B, node C can resume the DHCP transaction using this new relay. In this case, two possibilities can occur. One possibility is that the configuration process was interrupted before node C received a DHCP_OFFER message from a DHCP server. Then, node C has to send a new unicast DHCP_DISCOVER message to R_B. The other possibility is that node C has already received a DHCP_OFFER message from a DHCP server. Then, it can retrieve the offered IP parameters from the *addr_cache* and send a new DHCP_REQUEST message through DHCP relay R_B for the same IP parameters.

Note that the above-described polling mechanism is effective to quickly detect communication problems between the new node and the selected relay agent. However, if the DHCP transaction fails due to communication problems between the selected relay agent and its gateway, the polling mechanism is ineffective because the client will continue to receive the POLL_ACK messages. For this reason it is still necessary to implement a timeout to detect possible losses of DHCP messages. However, we substitute the legacy exponentially backoff algorithm used by DCHP clients to set retransmission timeouts with a fixed timeout T_A. We believe that the use of a fixed timeout is more suitable for a highly dynamic, and potentially lossy, environment, because it allows more prompt detection of failed DHCP transactions.

7. Experimental evaluation

To verify if our proposed scheme guarantees satisfactory address configuration delays and an acceptable efficiency in terms of protocol overheads, we have implemented a fully operational prototype of AH-DHCP, and we have tested it in a multi-hop WLAN, composed of two access points and five mobile nodes. To the best of our knowledge, stateful address autoconfiguration protocols (e.g., MANETconf), which are the schemes most similar to our approach, have been only validated via simulations, and no implementations are available. Note that publicly available solutions for address configurations in hybrid ad hoc networks and mesh networks are generally based on private addressing rather than routable Internet addresses, require NAT-based gateways, and use portions of the MAC addresses to build the internal IP address (see for instance, the addressing scheme used in MIT Roofnet [28] or in the Microsoft Mesh Connectivity Layer [29]). Thus, their functionalities are not comparable with our proposal. On the other hand, most experimental mesh networks use static addressing, while commercial mesh networks employ proprietary schemes.

For the sake of flexibility, we did not use commercial access points in our testbed, but computers equipped with both a wired and wireless interface, and implementing the gateway functionalities described in [5]. To develop the AH-DHCP prototype we adopted as reference implementation the DHCP client and relay agent public source code provided by the Internet System Consortium (ISC), which is one of the most popular DHCP distributions for POSIX-compliant operating systems [30]. Then, we made the necessary modifications to the DHCP software modules to implement the mechanisms described in Section 5. Concerning the DHCP server, we used the legacy DHCP server deployed on our campus wired network, which the gateways were attached to.

Regarding the hardware configuration, our testbed consists of seven *Acer Aspire 5633WLMi* laptops with *Intel Pro-Wireless 3945* as integrated wireless card. All nodes use a Linux 2.6.22 kernel and run the OLSR_Unik implementation in version 0.4.10, which is fully compliant with the RFC 3626 [27]. The ad hoc nodes are connected via IEEE 802.11*b* wireless links, transmitting at the maximum fixed rate of 11 Mbps. All nodes were located in the same room, and the *IP-tables* feature of Linux was used to emulate the multi-hop topologies. In our experiments, the background traffic is represented by persistent TCP flows, i.e., long-lived TCP connections transferring infinite-size files, and we used the *iperf* tool [31] to generate these flows.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

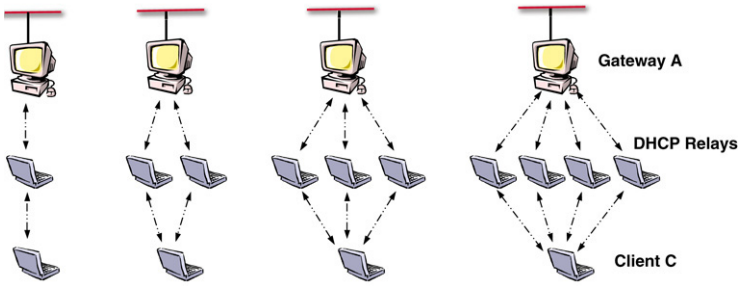


Fig. 4. Network layouts used for measuring the efficiency of DHCP relay discovery phase.

It is worth pointing out that we conducted the performance tests in an area of CNR building covered by other uncoordinated WLANs, which introduced uncontrollable radio interference. However, we believe that the randomness due to the external interference is well representing the characteristics of real radio environments and it is useful to attain more realistic results. To measure steady-state performance we have replicated each test two hundred times. The following graphs report both the average values and the 95% confidence intervals, which are generally very tight and not always easily appreciable from the graphs.

7.1. IP address configuration delay in static configurations

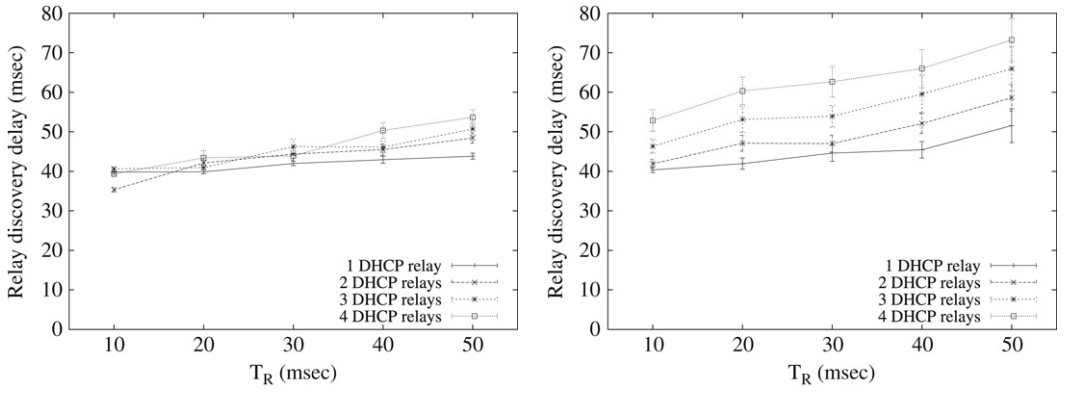
First, we carried out a set of experiments to select the most appropriate parameter setting for the DHCP relay discovery phase. Following the notation introduced in Section 6 and listed in Table 2, let T_R be the repetition period of RELAY_DISCOVER messages, and T_O the observation interval during which the new node collects the RELAY_ACK messages sent by the neighboring relay agents. In general, the new node can have several neighboring nodes already part of the multi-hop WLAN. Hence, it is important for the client to discover all the possible relays in order to select the best one (e.g., the relay at a shortest distance from an access point). It is obvious that the efficiency of the DHCP relay discovery phase depends on how frequently the new node generates RELAY_DISCOVER messages, and for how long it collects the relays' replies. In principle, the shorter the T_R period, the faster should be the discovery process. However, the closer two consecutive RELAY_DISCOVER messages are, the higher the probability that RELAY_ACK messages generated by different relays collide. To investigate this effect we used the network layouts illustrated in Fig. 4. More precisely, we considered a single client C with n neighboring AH-DHCP relays. All these potential relays are in radio visibility with the same access point A. Thus, the distance between the client C and the access point A is two hops. In the experiments we varied the T_R parameter and we forced the client to execute a continuous DHCP relay discovery procedure. Then, we measured the minimum time needed to receive a RELAY_ACK message from all the available relays. We initially performed our test without background traffic, i.e., when OLSR routing messages and AH-DHCP messages are the only packets transmitted over the wireless links. Then, we replicated the test introducing background traffic consisting of asymptotic TCP uplink flows opened between each relay and the gateway. If not otherwise stated, the TCP payload size is 1024 bytes.

Fig. 5(a) shows the minimum T_O interval needed to discover all the available relays in a network without background traffic as a function of T_R and for various n values. From the experimental results we observe that the time needed to complete the DHCP relay discovery procedure slightly increases by increasing the T_R period and the number of relays to discover. In addition, even for $T_R = 10$ ms (that is the shortest repetition period of RELAY_DISCOVER considered in our tests), the minimum time needed to discover a single relay is about 40 ms. By inspecting the packet traces we found out that this is mainly due to two reasons. Firstly, RELAY_DISCOVER messages are broadcast frames that are not protected by layer-2 retransmissions. Thus, the transmission of these messages is unreliable and they can get lost in the wireless channel. Secondly, the generation of RELAY_ACK packets may be subject to a non-negligible delay because the AH-DHCP relay module has to read the node's routing table to fill in the list of attributes, which is delivered within each RELAY _ACK massage. The user-space function we adopted to access the internal routing table introduces up to 10 ms of delay.

We replicated the same tests adding TCP background traffic saturating the wireless links, and Fig. 5(b) reports the measured minimum T_O interval. As expected, the minimum time needed to discover all the neighboring relays increases by introducing background traffic because both collision probability and queuing delays increase. However, with $T_R = 20$ ms, it is still possible to discover four DHCP relays in less than 60 ms.

Thus, according to our results $T_R = 20$ ms is a reasonable tradeoff between the promptness of the DHCP relay discovery phase and the protocol efficiency (a detailed analysis of the AH-DHCP overhead is reported in Section 7.3). Thus, the experimental results shown in the rest of this paper have been obtained by fixing $T_R = 20$ ms. Regarding the T_O interval, we express its value as a function of the T_R value as follows:

$$T_O = m \cdot T_R + \Delta,$$



(a) Without background traffic.

(b) With background traffic.

Fig. 5. Minimum duration of DHCP relay discovery phase.

where m is the maximum number of RELAY_DISCOVER messages a new node can send during a single observation period, and Δ is a guard time introduced to absorb jitter effects. In the following tests, we set $\Delta = T_R/2$, if not otherwise stated.

The second set of experiments we carried out aims at evaluating the total IP address configuration delay, say D_{conf}, which is defined as the time interval from the instant when the new node sends the first RELAY_DISCOVER message, and the instant at which it receives the DHCP_ACK message with the committed IP configuration parameters. The D_{conf} delay can be divided into two main components: D_{disc} and D_{assign}. The first component D_{disc} expresses the time between the first RELAY_DISCOVER message sent by the AH-DHCP client running on the new node and the election (through the unicast DHCP_DISCOVER message sent by the AH-DHCP client to the selected relay) of the DHCP relay agent acting as unique initiator of the address configuration process. It is intuitive to note that $D_{\text{disc}} \geq T_O$. In general, D_{disc} will be longer than T_O only if node C has not received any RELAY_ACK message during the initial observation period, and it has to repeat the DHCP relay discovery procedure. The second component D_{assign} expresses the time between the DHCP relay activation and the reception of the DHCP_ACK message that concludes the IP address assignment. In other words, the D_{assign} value represents the duration of the DHCP transaction established between the AH-DHCP client and the legacy DHCP server. Several factors can affect this delay, including the processing delays introduced by relay agents and DHCP servers [32]. However, in a multi-hop WLAN system also the distance of the DHCP server from the new node plays a crucial role in determining the D_{assign} value. To estimate this component of the D_{conf} delay we performed several tests in the network scenarios illustrated in Fig. 6. More precisely, we considered a single client C that is n wireless hops far from the access point A. Thus, at least $n - 1$ relays are needed to establish this n-hop path between C and A. Obviously, each wireless hop adds its own medium access delay, processing delay and queuing delay. Similarly to the results shown in Fig. 5 we performed our tests both without background traffic and with background traffic. In this case, the background traffic consists of $n - 1$ asymptotic TCP flows opened from each relay to the gateway.

Fig. 7(a) and (b) show the IP address configuration delay without and with background traffic, respectively, as a function of the T_O value and for different n values. As expected, there is a clear dependence of the total configuration delay on the duration of the observation period, because D_{disc} increases almost linearly with T_O (graphs are omitted due to space limitations). Moreover, the D_{conf} value increases by increasing the number of hops needed to reach the gateway. This delay increase is not significant in the experiments without background traffic, while it is considerable with background traffic. This is due to the increment of queuing delays caused by the TCP packets that are buffered in the transmission queues of DHCP relay nodes. In fact, without background traffic, the network contention induced by control messages (i.e., OLSR and DHCP packets) is negligible and the transmission buffers are empty most of the time. Consequently, most of the delay accumulated along the path is due to the processing delays introduced by DHCP relay agents. On the contrary, with background traffic, the transmission buffers may store several TCP packets. However, the experimental results show that the proposed autoconfiguration protocol ensures reasonably small address configuration delays (shorter than 0.8 s) even when the new joining node is distant five hops from the gateway, and the network is fully loaded.

7.2. IP address configuration delay in mobile configurations

In this section we evaluate the impact of node mobility on the total address configuration delay. To this end, we consider three different network scenarios, which are illustrated in Fig. 8(a), (b) and (c). Specifically, Fig. 8(a) represents the case of a new node C with a single neighboring AH-DHCP relay, say R_A, which is two hops far from the closest gateway G_A. Thus, after the DHCP relay discovery phase node C necessarily selects R_A as the unique initiator of the address configuration

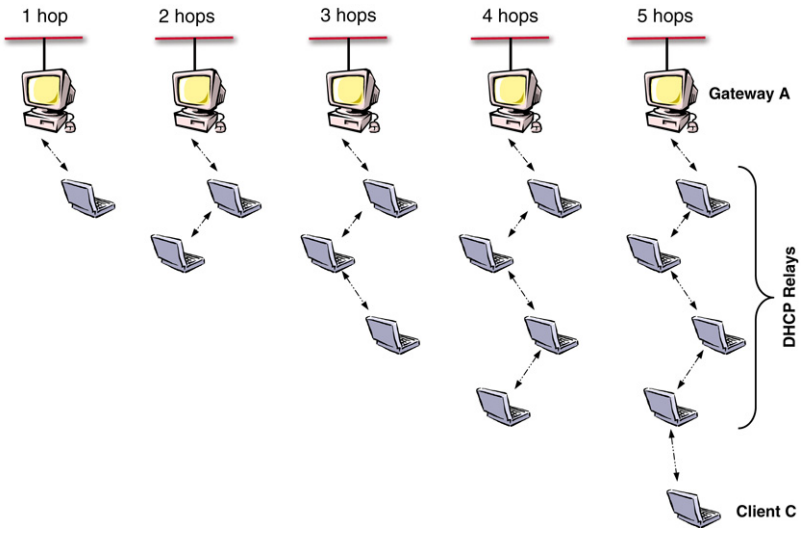
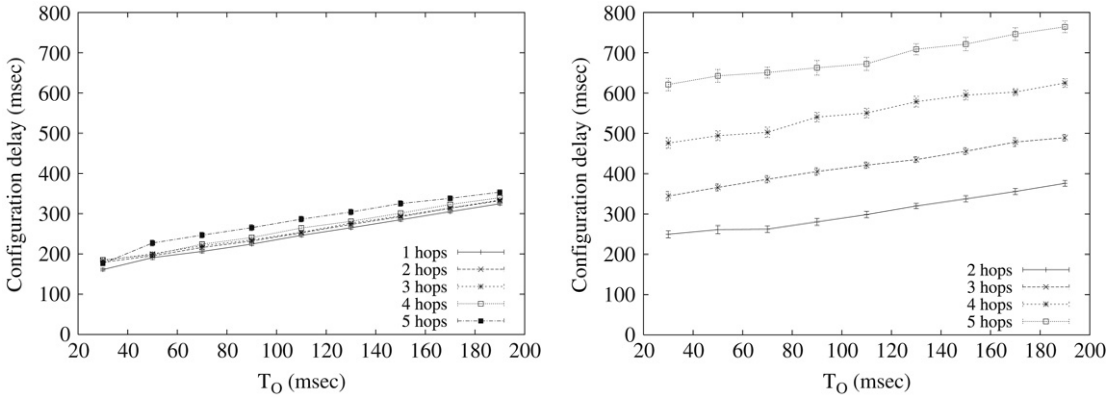


Fig. 6. Network layouts used for the measuring address configuration delays in static configurations.



(a) Without background traffic.

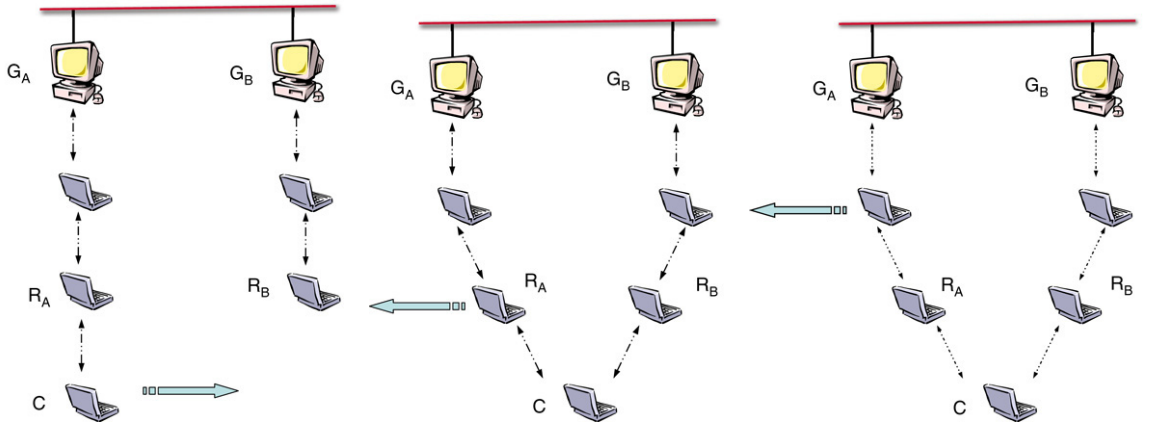
(b) With background traffic.

Fig. 7. IP address configuration delay in static chain topologies.

process. However, before completing the IP address assignment, node C moves out of node R_A's radio range. In this case, the polling mechanism allows a prompt detection of this event because node C stops receiving POLL_ACK messages from R_A. However, node C's *relay_cache* is empty and it has to trigger a new DHCP discovery phase to find another neighboring relay (i.e., node R_B). In Fig. 8(b) we illustrate a different case, because node C has now two neighboring AH-DHCP relay agents, both two hops away from a gateway. Therefore, after the DHCP relay discovery phase, node C will select randomly one of the two equivalent relays (R_A in our example) to start the address configuration process. Before completing the IP address assignment, node R_A moves out of node C's radio range. However, node C's *relay_cache* is not empty (it contains also the identity of relay R_B). Thus, node C can immediately start a new address configuration procedure. Finally, Fig. 8(c) illustrates a network scenario identical to Fig. 8(b), but in this case the mobile node is the intermediate node between R_A (the relay selected by node C in our example) and the gateway G_A. Since node R_A has lost its connectivity with the gateway, it stops replying to node C's polls. Note that R_A becomes aware of the topology change only when its link to the intermediate node expires.[4] After loosing its relay, node C will behave exactly as in the case illustrated in Fig. 8(b). For the sake of brevity, hereafter we denote the first scenario as *Scenario A*, the second one as *Scenario B*, and the last one as *Scenario C*. In what follows we report experimental results obtained by setting the period of RELAY_POLL messages equal to 50 ms, and the maximum number of consecutively missed POLL_ACK messages needed to declare a failed polling equal to four. This means that about 200 ms are needed by the polling scheme to declare lost a DHCP relay. Note that the T_A timeout is set to 3 s in our experiments, but the T_A value does not affect the system performance for the considered

[4] In our experiments, we configured OLSR to declare lost a link after 500 ms passed without receiving any OLSR message.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008



(a) Scenario A: New node C moves during DHCP transaction.

(b) Scenario B: AH-DHCP relay agent R_A moves during DHCP transaction.

(c) Scenario C: Intermediate node moves during DHCP transaction.

Fig. 8. Network layouts used for the measuring address configuration delays in mobile configurations.

mobility scenarios, where either the selected relay or the new node move while the other relays in the network are static.

Fig. 9(a) and (b) show the IP address configuration delays for all the three scenarios, without and with background traffic, respectively. Background traffic consists of two asymptotic TCP flows, one from node R_A to gateway G_A, and one from node R_B to gateway G_B. We set $T_R = 20$ ms, as in Section 7.1, and we investigated three representative values for the T_O parameter. In our tests, each mobile node (i.e., node C in Scenario A, node R_A in Scenario B, and the intermediate node between R_A and G_A in Scenario C) is configured to start moving after the completion of the DHCP relay discovery phase. Randomness is introduced in our experiments by inserting a random delay (uniformly selected in the range [50, 100] ms) between the completion of the DHCP relay discovery phase and the beginning of node's movement. As a result of this randomization, the DHCP transaction can be interrupted either before node C receives a DHCP_Offer message or before it receives the final DHCP_Ack message.

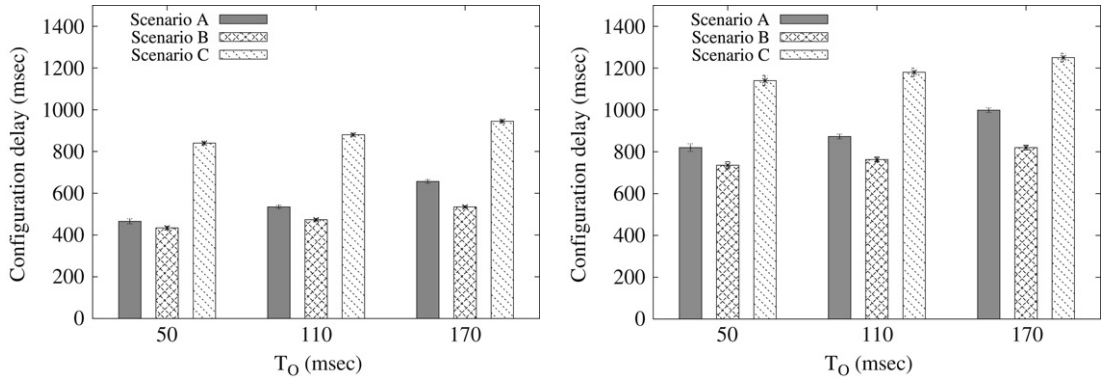
The experimental results shown in Fig. 9 indicate that, in the considered network scenarios, the address configuration delays are acceptable (always less than one second in Scenario A and Scenario B) and the polling mechanism ensures a prompt detection of relay unavailability. We can observe that in Scenario A the configuration delay is longer than the one measured in Scenario B. To explain this behavior we should note that, in the former case, node C has to perform at least two DHCP discovery phases, while in the latter case one DHCP discovery phase may be sufficient, because in Scenario B the *relay_cache* contains the identity of both node R_A and node R_B. Consequently, the longer the T_O interval, the more significant the delay difference between Scenario A and Scenario B. Regarding Scenario C, we can observe that the configuration delays are significantly higher than in the other two cases. The reason is that R_A becomes aware of the movement of the node that it is using as next-hop towards the gateway only after the OLSR link timeout. In our tests, this timeout is set to 500 ms, which corresponds to the difference in configuration delays between Scenario B and Scenario C. However, this additional delay is independent of our address configuration process, and it is only related to the dynamics of the ad hoc routing protocol.

Our experimental measurements show that background traffic negatively affects the address autoconfiguration process, which is an expected result that reproduces the behaviors observed also in static configurations (see Fig. 7). Finally it is worth pointing out that the use of a temporary *addr_cache* helps to reduce the configuration delays in case of mobility, especially for Scenario B. More precisely, if the DHCP transaction is interrupted after node C has received a DHCP_Offer message from a DHCP server, node C can resume the DHCP transaction by sending a new DHCP_Request message for the same IP parameters (which are stored in the *addr_cache*) through a new relay (see Fig. 2). This optimization avoids replicating the entire DHCP transaction after each topology change.

7.3. AH-DHCP protocol overheads

In the previous sections we focused on estimating the address configuration delays. However, another performance figure particularly relevant for an autoconfiguration protocol is the amount of protocol overheads generated. To evaluate this aspect we analyzed the size of all the packets transmitted and received from the new node when joining the multi-hop WLAN. Then, we classified the protocol overheads into three categories representing the packets generated and received during the DHCP relay discovery phase, the DHCP transaction and the DHCP relay polling. Figs. 10 and 11 show the protocol overheads in terms of bytes generated by/received from the end host for a few representative cases chosen from the network

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008



(a) Without background traffic.

(b) With background traffic.

Fig. 9. IP address configuration delay measured in mobile scenarios.

scenarios illustrated in Fig. 6 and in Fig. 8, respectively. On the other hand, Tables 3 and 4 report the protocol overheads in terms of number of packets generated by/received from the end host.

Our results indicate that, when analyzing the overheads in terms of bytes, the DHCP messages exchanged during the DHCP transaction are the dominant protocol overheads, and that the overall AH-DHCP overheads are practically negligible (less than 1 kbyte). This can be explained by observing that the payload of AH-DHCP control packets (i.e., RELAY_DISCOVER, RELAY_ACK, RELAY_POLL and POLL_ACK messages) are 44-byte long (28 bytes for the IP and UDP headers plus 16 bytes for the payload listing the node's attributes), while DHCP packets, in our DHCP version, are either 1472-byte long (DHCP_DISCOVER and DHCP_REQUEST messages) or 300-byte long (DHCP_OFFER and DHCP_ACK messages). On the contrary, if we analyze the protocol overheads in terms of packets, we can observe that the overhead associated to the DHCP transaction is the smallest one, only four packets, and it is independent of both the specific setting for AH-DHCP parameters, and the network topology. However, it is worth pointing out that DHCP messages are replicated on each wireless hop they traverse on the path between the selected relay and the closest gateway. This means that to compute the overall protocol overheads due to DHCP messages, the overheads reported in Table 3 should be multiplied by n, where n is the hop distance between the selected relay and the closest gateway. In any case, the DHCP server will receive only one copy of each DHCP message.

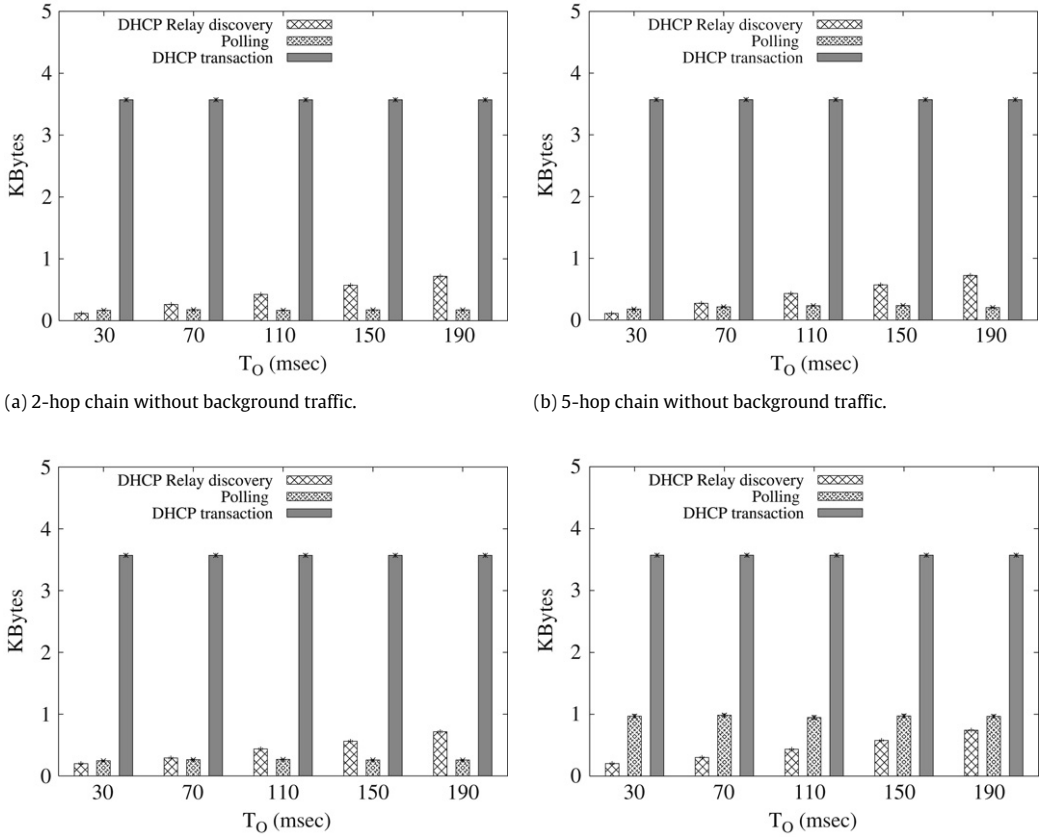
As expected, the AH-DHCP overheads generated during the DHCP relay discovery phase depend on the T_O value. Specifically, the longer the T_O value, the more RELAY_DISCOVER messages are generated, and the more RELAY_ACK messages are received by node C. For instance, let us consider the case $T_O = 70$ ms in Table 3. Since $T_R = 20$ ms, node C sends three RELAY_DISCOVER messages and, in principle, it should receive three RELAY_ACK messages.[5] In our test conditions, the link quality is good and messages are rarely lost due to channel noise. Thus, the measured overhead is very close to the expected value of six packets. Note that, with background traffic the overhead increases rather than decreasing. The explanation of this behavior is that a single DHCP relay discovery phase is not always sufficient to node C to discover its DHCP relay.

As shown in Figs. 10 and 11, the AH-DHCP overheads generated during the DHCP relay polling are independent of the T_O value, but are affected by the presence of background traffic and the number of hops between the new node and the gateway. This is easily explained by noting that the duration of the DHCP transaction (i.e., D_{assign}) increases when the distance between the new node and the gateway increases, especially if background traffic disturbs the DHCP transaction (see Fig. 7). Thus, the longer the D_{assign}, the more the RELAY_POLL messages are generated, and the more POLL_ACK messages are received by node C. From the shown results it is evident that AH-DHCP overheads in terms of packets are significantly higher than DHCP overheads. However, this overhead can be reduced by adjusting the repetition periods of RELAY_DISCOVER and RELAY_POLL messages. In addition, the number of generated messages is quite low and it is reasonable to believe that it has no negative impact on the access delay of data packets.

Similar considerations can be derived by analyzing Fig. 11. The main difference we can notice is that the protocol overheads generated by the DHCP transactions in Scenario A are higher than the ones generated in Scenario B and Scenario C. This can be explained by observing that in Scenario B and Scenario C, if the mobile node moves after node C has received a DHCP_OFFER message, then node C can resume the DHCP transaction by sending a new DHCP_REQUEST message directly to relay R_B. On the contrary, in Scenario A node C always has to restart a completely new DHCP transaction after losing the radio visibility with relay R_A. Therefore, a higher number of DHCP messages are generated in Scenario A than Scenario B, resulting in higher protocol overheads. However, as reported in Table 4, less than two DHCP messages have to

[5] Node C may receive less than three RELAY_ACK messages either because the RELAY_ACK messages are lost due to channel noise/ contention, or because the relay did not receive the RELAY_DISCOVER message.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008



(a) 2-hop chain without background traffic.

(b) 5-hop chain without background traffic.

(c) 2-hop chain with background traffic.

(d) 5-hop chain with background traffic.

Fig. 10. AH-DHCP protocol overheads (in bytes) for the network scenarios illustrated in Fig. 6.

Table 3
AH-DHCP protocol overheads (in packets) for the network scenarios illustrated in Fig. 9.

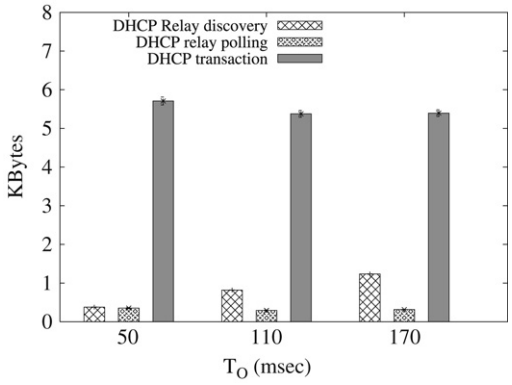
T_O (ms)		2-hop chain		5-hop chain	
		w/o back. traffic	with back. traffic	w/o back. traffic	with back. traffic
70	DHCP relay discovery	5.95	6.72	5.96	7
	Polling	4.08	6.11	4.95	18.2
	DHCP transaction	4	4	4	4
150	DHCP relay discovery	13.02	13.29	13.26	13.40
	Polling	4.02	6	5.47	20.4
	DHCP transaction	4	4	4	4

be retransmitted, on average, to complete the DHCP transaction. In addition, we can observe that the polling overhead is maximum for Scenario C because relay R_A keeps replying to the RELAY_POLL messages until the ad hoc routing protocol does not declare lost its connection to the gateway. On the other hand, Scenario A has the highest overhead for the DHCP relay discovery phase because at least two separate discovery procedures are necessary to discover relay R_A and relay R_B, while in both Scenario B and Scenario C the two relay are discovered during the first initial DHCP discovery phase.

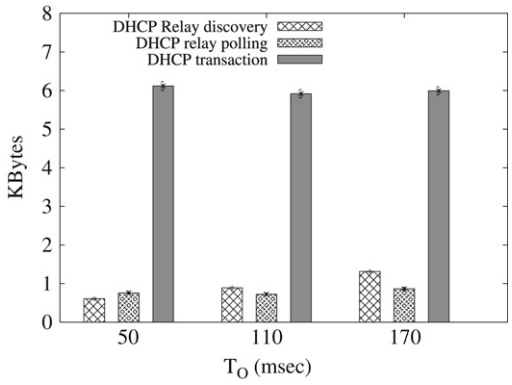
8. Conclusions

In this paper we described AH-DHCP, an address autoconfiguration protocol for multi-hop WLAN. The main goal of our work was to prove the applicability of DHCP, originally designed to provide configuration parameters to hosts in a fixed network, also when traditional WLANs integrate ad hoc networking technologies to discover and maintain multi-hop wireless path within the network. The basic idea was to take advantage of DHCP relay capabilities available in already configured nodes. To this end, we proposed extensions to DHCP to enable a new node to dynamically choose a reachable relay

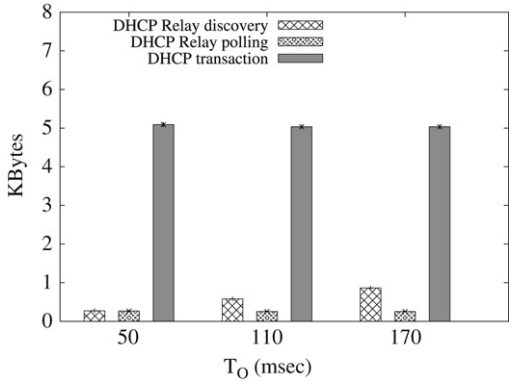
Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008



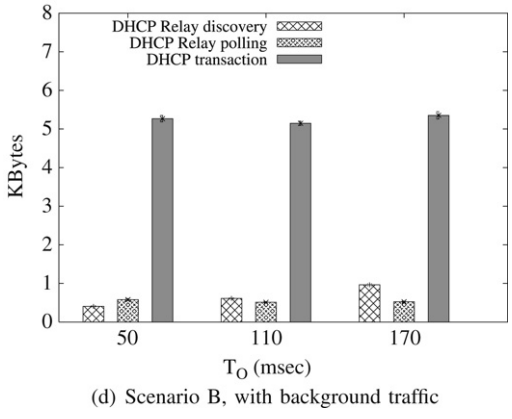
(a) Scenario A, without background traffic.



(b) Scenario A, with background traffic.

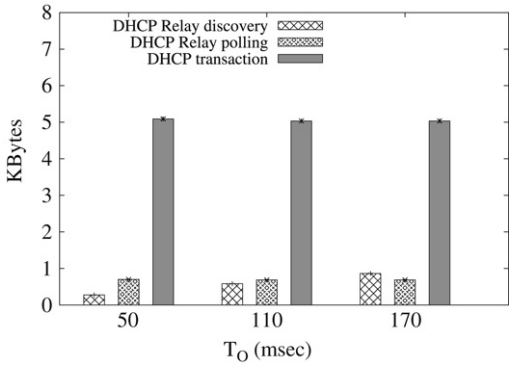


(c) Scenario B, without background traffic.

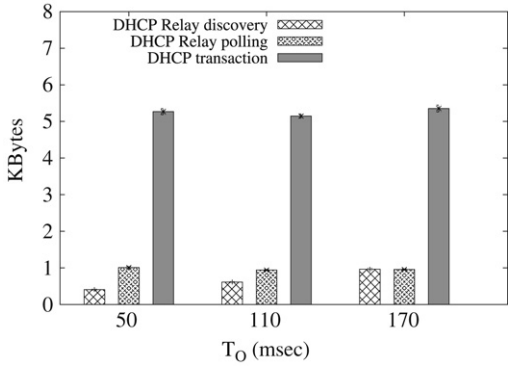


(d) Scenario B, with background traffic

(d) Scenario B, with background traffic.



(e) Scenario C, without background traffic.



(f) Scenario C, with background traffic.

Fig. 11. AH-DHCP protocol overheads (in bytes) for the network scenarios illustrated in Fig. 9.

Table 4
AH-DHCP protocol overheads (in packets) for the network scenarios illustrated in Fig. 11 for $T_O = 70$ ms.

		Scenario A	Scenario B	Scenario C
DHCP relay discovery	w/o back. traffic	19.1	13.52	13.6
	with back. traffic	20.6	14.22	14.27
Polling	w/o back. traffic	6.8	5.95	16
	with back. traffic	16.8	11.8	21.86
DHCP transaction	w/o back. traffic	6.02	5.64	5.75
	with back. traffic	6.62	5.76	5.90

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

agent as the unique initiator of the configuration procedure. Then, this relay transparently passes all the client-originated messages to the DHCP servers located in the wired part of the network. Our proposed solution can tolerate message losses and node mobility because it implements appropriate mechanisms to promptly react to persistent communication problems and topology changes.

Experiments conducted with a prototype implementation of AH-DHCP have shown that our solution ensures short address configuration delays and low protocol overheads, even when node mobility or background traffic interferes with the operations of the autoconfiguration protocol. For future work, we intend to investigate mechanisms to reduce the impact of multi-hop forwarding on address assignment delays in large-scale multi-hop WLANs, e.g., by introducing a hierarchy of DHCP relay agents. Another possible research direction is the extension of our solution to IPv6.

Acknowledgement

This work was supported in part by the European Commission through project EU-MESH (Enhanced, Ubiquitous, and Dependable Broadband Access using MESH Networks), FP7 ICT-215320.

References

[1] Local and metropolitan area network – specific requirements – Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, The Institute of Electrical and Electronics Engineer, Piscataway, NJ, August. 1999.
[2] S. Lee, S. Banerjee, B. Bhattacharjee, The case for a multi-hop wireless local area network, in: Proc. of IEEE INFOCOM 2004, vol. 2, Hong Kong, China, March 7–11 2004, pp. 894–905.
[3] R. Karrer, A. Sabharwal, E. Knightly, Enabling large-scale wireless broadband: The case for TAPs, ACM SIGMOBILE Comput. Comm. Review 34 (1) (2004) 27–34.
[4] S. Narayanan, P. Liu, S. Panwar, On the advantages of multi-hop extensions to the IEEE 802.11 infrastructure mode, in: Proc. of IEEE WCNC 2005, vol. 1, New Orleans, LA, USA, March 13–17 2005, pp. 132–138.
[5] E. Ancillotti, R. Bruno, M. Conti, E. Gregori, A. Pinizzotto, A layer-2 framework for interconnecting ad hoc networks to fixed internet: Test-bed implementation and experimental evaluation, Comput. J. 50 (4) (2007) 478–499.
[6] M. Conti, S. Giordano, Multihop ad hoc networking: The theory, IEEE Commun. Mag. 45 (4) (2007) 78–86.
[7] Y.-D. Lin, Y.-C. Hsu, Multihop cellular: A new architecture for wireless communications, in: Proc. of IEEE INFOCOM 2000, vol. 3, Tel Aviv, Israel, March 26–30 2000, pp. 1273–1282.
[8] R. Bruno, M. Conti, E. Gregori, Mesh networks: Commodity multihop ad hoc networks, IEEE Commun. Mag. 43 (3) (2005) 123–131.
[9] R. Droms, Dynamic host configuration protocol, RFC 2131, March 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2131.txt.
[10] S. Cheshire, B. Aboba, E. Guttman, Dynamic configuration of IPv4 link-local addresses, RFC 3927, May 2005. [Online]. Available: http://www.ietf.org/rfc/rfc3927.txt.
[11] R. Bruno, M. Conti, A. Pinizzotto, Enhancing DHCP for address autoconfiguration in multi-hop wlans, in: ICDCN 2008, in: Lecture Notes in Computer Science, vol. 4904, Springer, Kolkata, India, 2008, pp. 528–539.
[12] K. Weniger, M. Zitterbart, Address autoconfiguration on mobile ad hoc networks: Current approaches and future directions, IEEE Network 18 (4) (2004) 6–11.
[13] P. Engelstad, G. Egeland, NAT-based internet connectivity for on demand MANETs, in: Proc. of WONS 2004, Madonna di Campiglio, Italy, January 18–23 2004, pp. 4050–4056.
[14] P. Engelstad, A. Tønnesen, A. Hafslund, G. Egeland, Internet connectivity for multi-homed proactive ad hoc networks, in: Proc. of IEEE ICC'2004, vol. 7, Paris, France, June 20–24 2004, pp. 4050–4056.
[15] S. Nesargi, R. Prakash, MANETconf: Configuration of hosts in a mobile ad hoc network, in: Proc. of IEEE INFOCOM 2002, vol. 2, New York, NY, USA, June 23–27 2002, pp. 1059–1068.
[16] H. Zhou, L. Ni, M. Mutka, Prophet address allocation for large scale MANETs, Ad Hoc Netw. J. 1 (4) (2003) 423–434.
[17] C. Perkins, J. Malinen, R. Wakikawa, E. Belding-Royer, Y. Sun, IP address autoconfiguration for ad hoc networks, Internet Draft, July 2002.
[18] N. Vaidya, Weak duplicate address detection in mobile ad hoc networks, in: Proc. of ACM MobiHoc 2002, Lausanne, Switzerland, June 9–11 2002, pp. 206–216.
[19] K. Weniger, PACMAN: Passive autoconfiguration for mobile ad hoc networks, IEEE JSAC 23 (3) (2005) 507–519.
[20] IETF, Ad-Hoc Network Autoconfiguration (autoconf) WG, December 2007. [Online]. Available: http://www.ietf.org/html.charters/autoconf-charter.html.
[21] S. Thomson, T. Narten, IPv6 stateless address autoconfiguration, RFC 2462, December 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2462.txt.
[22] Z. Fan, IPv6 stateless address autoconfiguration in ad hoc networks, in: Proc. of PWC'03, Venice, Italy, September 23–25 2003, pp. 665–678.
[23] R. Wakikawa, J. Malinen, C. Perkins, A. Nilsson, A. Tuominen, Global connectivity for IPv6 mobile ad hoc networks, Internet Draft, March 2006. [Online]. Available: http://tools.ietf.org/wg/manet/draft-wakikawa-manet-globalv6-05.txt.
[24] C. Jelger, T. Noel, A. Frey, Gateway and address autoconfiguration for IPv6 ad hoc networks, Internet Draft, April 2004. [Online]. Available: http://clarinet.u-strasbg.fr/~frey/draft-jelger-manet-gateway-autoconf-v6-02.txt.
[25] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney, Dynamic host configuration protocol for IPv6 (DHCPv6), RFC 3315, July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3315.txt.
[26] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, in: Proc. of ACM MobiCom'99, Seattle, WA, USA, August 15–20 1999, pp. 151–162.
[27] T. Clausen, P. Jaquet, Optimized link state routing protocol (OLSR), RFC 3626, October 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt.
[28] J. Bicket, D. Aguayo, S. Biswas, R. Morris, Architecture and evaluation of an unplanned 802.11b mesh network, in: Proc. of ACM MobiCom, Cologne, Germany, August 28–September 2 2005, pp. 31–42.
[29] R. Draves, J. Padhye, B. Zill, Routing in Multi-radio, multi-hop wireless mesh networks, in: Proc. of ACM MobiCom, Philadelphia, PE, USA, September 26–Oct. 1 2004, pp. 114–128.
[30] Internet systems consortium, ISC DHCP Version 3.0.5, November 5 2006. [Online]. Available: http://www.isc.org/index.pl?/sw/dhcp/.
[31] NLANR/DAST, Iperf Version 2.0.2, May 3 2005. [Online]. Available: http://dast.nlanr.net/Projects/Iperf/.
[32] A. Park, P. Kim, M. Lee, Y. Kim, Fast address configuration for WLAN, in: Parallel and Distributed Computing: Applications and Technologies, in: Lecture Notes in Computer Science, Springer, 2004, pp. 396–400.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008

Emilio Ancillotti received the Laurea degree in Computer Engineering, and Ph.D. degree in Information Engineering, both from the University of Pisa, in 2003 and 2007, respectively. From 2003 to 2006 he worked as the teaching assistant at the Department of Information Engineering of the University of Pisa. After the Ph.D., from 2007, is a researcher at IIT, an institute of the Italian National Research Council (CNR). His current research interests are in the area of wireless and mobile networks with an emphasis on pervasive computing, ad hoc networks, mesh networks and performance evaluation.

Raffaele Bruno is a researcher at IIT, an institute of the Italian National Research Council (CNR). He received Ph.D. in computer engineering and the Laurea in telecommunication engineering, both from the University of Pisa, Italy, in 2003 and 1999, respectively. His main research interests include the design, modelling and performance evaluation of MAC, routing and transport protocols for wireless networks. He published more than 30 papers in international journals, conferences and workshops. He served and is currently serving on the TPC of several international conferences and workshops. He was the Workshop Co-Chair of *IEEE PerSeNS* 2006 and *IEEE MASS-GHS*07. He is currently guest editor for a Special Issue of the Pervasive and Mobile Computing (PMC) Journal on Homeland and Global Security.

Marco Conti is a research director at IIT, an institute of the Italian National Research Council (CNR). He co-authored the book "Metropolitan Area Networks" (1997) and is the co-editor of the books "Mobile Ad Hoc Networking" (2004) and "Mobile Ad Hoc Networks: From Theory to Reality" (2007). He published in journals and conference proceedings more than 200 research papers related to design, modeling, and performance evaluation of computer-network architectures and protocols. He served as TPC chair of *IEEE PerCom* 2006, and of the IFIP-TC6 Conferences *Networking* 2002 and *PWC* 2003, and as TPC co-chair of *ACM WoWMoM* 2002, *WiOpt* '04, *IEEE WoWMoM* 2005, and *ACM MobiHoc* 2006. He served as general chair of *ACM REALMAN* 2006 and IEEE MASS 2007, and as general co-chair of *IEEE WoWMoM* 2006, and *ACM MobiOpp* 2007. He is Associate Editor-in-Chief of *Pervasive and Mobile Computing* Journal, and he is on the editorial board of: *IEEE Transactions on Mobile Computing, Ad Hoc Networks* journal and *Wireless Ad Hoc and Sensor Networks: An International Journal.*

Antonio Pinizzotto is a researcher at IIT, an institute of the Italian National Research Council (CNR). He received the Laurea degree in Electronic Engineering from the University of Pisa, Italy, in 1992. He initially worked on experimental deployments of various technologies applied to computer networks, including quality of service, VoIP, multicast, network monitoring, IPv6. His current research interests are in the area of wireless mesh networks, with special emphasis on performance analysis, QoS-based routing, and protocols for the interconnection of heterogeneous networks.

Please cite this article in press as: E. Ancillotti, et al., Dynamic address autoconfiguration in hybrid ad hoc networks, Pervasive and Mobile Computing (2008), doi:10.1016/j.pmcj.2008.09.008